Available online at www.sciencedirect.com

ScienceDirect

International Journal of Approximate Reasoning
45 (2007) 255–270

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

www.elsevier.com/locate/ijar

On-line alert systems for production plants: A conflict based approach

Thomas D. Nielsen *, Finn V. Jensen

Department of Computer Science, Aalborg University, Fredrik Bajers vej 7E, 9220 Aalborg Ost, Denmark

Received 22 December 2005; received in revised form 9 June 2006; accepted 30 June 2006
Available online 11 August 2006

Abstract

We present a new methodology for detecting faults and abnormal behavior in production plants. The methodology stems from a joint project with a Danish energy consortium. During the course of the project we encountered several problems that we believe are common for projects of this type. Most notably, there was a lack of both knowledge and data concerning possible faults, and it therefore turned out to be infeasible to learn/construct a standard classification model for doing fault detection. As an alternative we propose a method for doing on-line fault detection using only a model of normal system operation. Faults are detected by measuring the conflict between the model and the sensor readings, and knowledge about the possible faults is therefore not required. We illustrate the proposed method using real-world data from a coal driven power plant as well as simulated data from an oil production facility.

Keywords: Alert systems; Bayesian networks; Conflict measure

1. Introduction

Most production plants are equipped with sensors providing information to a control room where operators monitor the production process. Based on skill and experience the operators are alerted if something unusual happens, and through inspection of sensor

* Corresponding author. Tel.: +45 9635 9854; fax: +45 9815 9889.
E-mail addresses: tdn@cs.aau.dk (T.D. Nielsen), fvj@cs.aau.dk (F.V. Jensen).

0888-613X/$ - see front matter
doi:10.1016/j.ijar.2006.06.010

readings, or derivatives thereof (so-called soft sensors), a diagnostic process may be initiated.

In connection to a joint project with an energy consortium, we have been working on establishing an alert system for a coal driven power plant. By an alert system we mean a system that, based on sensor readings, raises a flag in case of an abnormal situation. We intended to base the system on a Bayesian network representation [19,13] of the power plant, and to help us establish the model we had access to process engineers and an extensive database of logged sensor data. However, during the course of the project we encountered several problems, which we believe are common for projects of this type:

(1) The engineers' knowledge of the plant is not sufficient for providing a causal structure.
(2) The production process is so complex that it is difficult for the engineers to specify the possible faults (abnormal situations) and, in particular, how these faults would manifest themselves in the sensor readings.
(3) The time constants, describing the delay from event to effect, are difficult to determine.
(4) Faults are so rare that statistics cannot be used to learn neither the structure nor the parameters of a model of the faults.
(5) As there is a difference between a true value and its sensor reading, true values should appear as hidden variables.

Faced with these problems, one approach would be to get as much causal structure from the engineers as possible and to combine this information with a data driven learning method. Unfortunately, state of the art of structural learning algorithms cannot cope with domains with a massive set of hidden variables. Furthermore, due to the lack of knowledge about the possible faults it is not obvious how such a model should subsequently be used for classifying abnormal behavior; as done in e.g., [4,18].

In this paper we propose an alternative methodology for on-line detection of abnormal behavior in production systems. The method focuses on systems which are prone to the problems described above, and it has the desirable property that it does *not* require information about the possible faults nor a model of abnormal behavior. We illustrate the proposed method using real-world data from the above mentioned power plant as well as simulated data from an oil production facility.

2. The proposed methodology

As implied above, it is not obvious how to construct a classifier (encoding the possible faults) for detecting abnormal behavior; neither in the form of a causal model nor in the form of e.g., a Naïve Bayes model [9] or a tree augmented Naïve Bayes model [10].

Instead, we propose to learn a Bayesian network representing normal operation only. At each time step the model is used to calculate the probability of the set of sensor readings for that time step. This probability is in turn used to evaluate whether the sensor readings are jointly outside the scope of normal operation. That is, the proposed methodology consists of two steps: (i) learning a model of the sensors for normal operation, and (ii) using the learned model to monitor the system, initiate alerts and perform on-line diagnostics. Models for describing normal operation has also been explored in the model-based

diagnosis community [20,8]: based on a pre-specified model of normality (formulated in first-order logic), each component in the system is assigned a state (either normal or abnormal) which is consistent with both the model and any observations made of the system.

2.1. Learning a model

The available database consists of sensor readings that have been logged during normal system operation; each instance in the database can be seen as a "snapshot" of the overall production process. In what follows we shall assume that the production process is composed of a temporally ordered collection $(C_1, C_2, \ldots, C_n)$ of components (or sub-processes). The output of component C_i serves as input to component C_{i+1}, and (for ease of exposition) each component, C_i, is assumed to be equipped with a single sensor, S_i. For instance, when tracking the coal in a power plant we can, at an abstract level, describe the overall production process as being composed of three components: the silo, the coal mill, and the furnace. Since the production process is a physical non-instantaneous process we also have a delay (or time constant) associated with each of the components, i.e., the time it takes for a particular unit (e.g., a piece of coal) to pass through.

Based on this perspective, we initially considered learning a model of the flow of one unit (e.g., coal) through the production plant. The variables in the learned model would then represent the sensors in the system. One approach for learning such a model would be to first transform the original database s.t. a case in the transformed database would correspond to the sensor readings related to one particular unit (this transformation is illustrated in Table 1). However, making such a transformation requires information about the time constants, which was unfortunately not available.

An alternative approach would be to learn a dynamic Bayesian network model directly from the database by treating the cases as representing a trajectory through the system

Table 1
The original database is transformed s.t. each case in the resulting database contains the sensor readings related to one particular unit in the system. We have assumed that the time delay between sensor S_1 and S_2 corresponds to the sampling delay between case/snapshot $\boldsymbol{c}_1$ and $\boldsymbol{c}_j$ in the original database

	S_1	S_2	$\cdots$	S_n
$\boldsymbol{c}_1$	$\underline{x_1^1}$	x_2^1	$\cdots$	x_n^1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\boldsymbol{c}_j$	x_1^j	$\underline{x_2^j}$	$\cdots$	x_n^j
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\boldsymbol{c}_k$	x_1^k	x_2^k	$\cdots$	$\underline{x_n^k}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\boldsymbol{c}_N$	x_1^N	x_2^N	$\cdots$	x_n^N

$\Rightarrow$

	S_1	S_2	$\cdots$	S_n
$\boldsymbol{c}_1$	$\underline{x_1^1}$	$\underline{x_2^j}$	$\cdots$	$\underline{x_n^k}$
$\boldsymbol{c}_2$	x_1^2	x_2^l	$\cdots$	$x_n l'$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\boldsymbol{c}_k$	x_1^k	x_2^m	$\cdots$	$x_n^{m'}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\boldsymbol{c}_{N'}$	$x_1^{N'}$	$x_2^{N'}$	$\cdots$	x_n^N

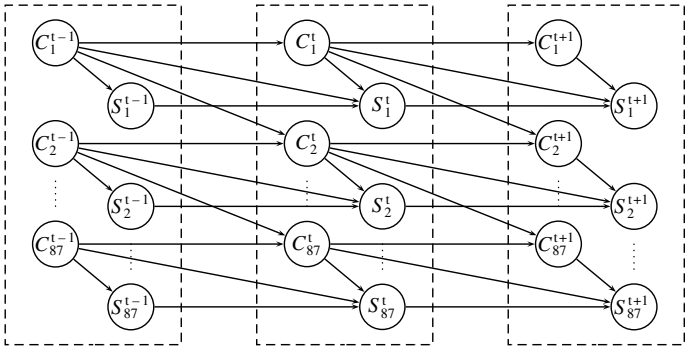

Fig. 1. The figure illustrates a dynamic Bayesian network representation of the data generation process for a production plant. The variable S_i represents the sensor associated with component C_i, and the arcs going into a sensor variable from a previous time slice models that the state of a sensor (correct, faulty or drifting) has an impact on the next sensor reading.

[11,1]. Unfortunately, learning such a model also requires information about the time constants.

Instead, we used the database directly to learn a Bayesian network model over the sensor variables. This approach, however, has a potential computational drawback in the sense that we must expect the learned model to be very dense (this was also confirmed in the empirical experiments). If we had known the time constants we would also have expected a dense model, albeit to a smaller degree. To see this, consider Fig. 1 which illustrates a simplified temporal causal model of the data generation process for a production plant. Learning a model for the sensor variables can now conceptually be seen as learning a model that describes the marginal distribution over the sensor variables S_i in a time slice. However, according to Fig. 1 we see that after very few time steps, each pair of variables in a time slice are dependent no matter how we condition on the other variables in the time slice. This is not only due to the hidden variables (modeling the components in the system), but also because standard learning methods treat the cases as being independent [5]; the latter corresponds to the past being unobserved.

2.2. Initiation of alerts

The sensor readings are received in a constant flow, which is chopped up into time steps of, say, 1 s. This means that for every second we have evidence consisting of a value for each variable in the model.

Let the evidence be $\boldsymbol{e} = \{e_1, \ldots, e_n\}$, where e_i is a sensor reading. We can now calculate the conflict measure for the evidence [14] as

$$\mathrm{conf}(\boldsymbol{e}) = \log\left[\frac{P(e_1)\cdot\ldots\cdot P(e_n)}{P(\boldsymbol{e})}\right].$$

Since the learned model represents normal system operation we would in general expect that sensor readings recorded during normal operation are positively correlated (i.e., $\mathrm{conf}(\boldsymbol{e}) \leqslant 0$) relative to the model. That is, for two sensor readings e_i and e_j we would expect $P(e_i|e_j) > P(e_i)$, and therefore $P(e_i, e_j) = P(e_i|e_j)P(e_j) > P(e_i)P(e_j)$. Thus, when $\mathrm{conf}(\boldsymbol{e}) > 0$

then this is an indication of an abnormal situation, and an alert may be triggered, see also [16,15]. The conflict measure can also be interpreted as a soft measure of inconsistency: if a case is inconsistent with the model, then it has probability 0, and if it is close to being inconsistent then it has an unusual low probability; "unusual" is for this measure calculated relative to the model for complete independence. For the conflict measure above, we expect a rather constant level for conf(·) under stable normal operation. When the process is changed, and it transforms from one mode of normal operation to another, we should expect oscillations in the conflict values until the changes have propagated and resulted in a new stable mode of normal operation.

The probabilities $P(e_i)$ can be read directly from the Bayesian network in its initial state, and it does not require any propagation. As all variables in the model are instantiated, $P(\boldsymbol{e})$ is also very easy to calculate: it is simply the product of the appropriate entries in the conditional probability tables of the Bayesian network. No propagation is required, i.e., the complexity is linear in the number of variables in the model.

As noted above, a positive conflict value is an indication of an abnormal situation. On the other hand, a negative conflict value does not necessarily imply that we have a normal situation as it may hide a serious conflict: if the sensors are strongly correlated during normal operation, the conflict level will be very negative, and a few conflicting sensor readings may therefore not cause the entire conflict to be positive. This can also be seen from the following proposition.

Proposition 2.1. *Let* $\boldsymbol{e}^x = \{e_1^x, \ldots, e_n^x\}$, $\boldsymbol{e}^y = \{e_1^y, \ldots, e_m^y\}$, *and* $\boldsymbol{e} = \boldsymbol{e}^x \cup \boldsymbol{e}^y$. *Then*

$$\operatorname{conf}(\boldsymbol{e}) = \operatorname{conf}(\boldsymbol{e}^x, \boldsymbol{e}^y) + \operatorname{conf}(\boldsymbol{e}^x) + \operatorname{conf}(\boldsymbol{e}^y),$$

where $\operatorname{conf}(\boldsymbol{e}^x, \boldsymbol{e}^y) = \log\left[\frac{P(\boldsymbol{e}^x)P(\boldsymbol{e}^y)}{P(\boldsymbol{e})}\right]$.

Proof

$$\begin{aligned}
\operatorname{conf}(\boldsymbol{e}) &= \log\left[\frac{P(e_1^x)\cdot\ldots\cdot P(e_n^x)\cdot P(e_1^y)\cdot\ldots\cdot P(e_m^y)}{P(\boldsymbol{e})}\right] \\
&= \log\left[\frac{P(e_1^x)\cdot\ldots\cdot P(e_n^x)\cdot P(e_1^y)\cdot\ldots\cdot P(e_m^y)P(\boldsymbol{e}^x)P(\boldsymbol{e}^y)}{P(\boldsymbol{e})P(\boldsymbol{e}^x)P(\boldsymbol{e}^y)}\right] \\
&= \log\left[\frac{P(\boldsymbol{e}^x)P(\boldsymbol{e}^y)}{P(\boldsymbol{e})}\cdot\frac{P(e_1^x)\cdot\ldots\cdot P(e_n^x)}{P(\boldsymbol{e}^x)}\cdot\frac{P(e_1^y)\cdot\ldots\cdot P(e_m^y)}{P(\boldsymbol{e}^y)}\right] \\
&= \operatorname{conf}(\boldsymbol{e}^x, \boldsymbol{e}^y) + \operatorname{conf}(\boldsymbol{e}^x) + \operatorname{conf}(\boldsymbol{e}^y). \qquad \square
\end{aligned}$$

So, it may happen that $\boldsymbol{e}^x$ and $\boldsymbol{e}^y$ are internally so strongly correlated that they dominate a conflict between the two sets. Thus, even when the conflict is negative, we shall watch out for jumps in the conflict level that may indicate a potential abnormal situation.

When an alert has been triggered, the system can start tracing the source of the alert. Various ways of tracing the conflict may be used. In our case we perform a greedy *conflict resolution*: recursively remove the sensor reading that reduces the conflict the most, and continue until the conflict is below a predefined threshold. This procedure can be performed very fast by exploiting fast retraction [7], lazy propagation [17] or arithmetic circuits [6] as can be seen from the following proposition.

Proposition 2.2. *Let* $\boldsymbol{e} = \{e_1, \ldots, e_n, e_x\}$ *be evidence,* X *a variable with evidence* e_x*, and* $\boldsymbol{e}^{-x}$ *the remaining evidence. Then*

$$\mathrm{conf}(\boldsymbol{e}) = \log\left[\frac{P(e_x)}{P(e_x|\boldsymbol{e}^{-x})}\right] + \mathrm{conf}(\boldsymbol{e}^{-x}).$$

Proof

$$\begin{aligned}\mathrm{conf}(\boldsymbol{e}) &= \log\left[\frac{P(e_x)P(e_1)\cdot\ldots\cdot P(e_n)}{P(\boldsymbol{e})}\right] = \log\left[\frac{P(e_x)}{P(e_x|\boldsymbol{e}^{-x})}\cdot\frac{P(e_1)\cdot\ldots\cdot P(e_n)}{P(\boldsymbol{e}^{-x})}\right]\\ &= \log\left[\frac{P(e_x)}{P(e_x|\boldsymbol{e}^{-x})}\right] + \mathrm{conf}(\boldsymbol{e}^{-x}). \qquad \square\end{aligned}$$

That is, the reading with lowest normalized likelihood given the other readings contributes the most to the conflict. Note that as the Markov blanket of X is instantiated, the calculation of $P(e_x|\boldsymbol{e}^{-x})$ can be performed locally.

Algorithm 2.1 (*Conflict resolution*)

(1) Let t be a "conflict threshold" for when an alert should be initiated (e.g., $t = 0$).
(2) Let $\boldsymbol{e}$ be a set of conflicting sensor readings.
(3) Repeat
    (a) Select

$$e' = \arg\max_{e} \log\left[\frac{P(e)}{P(e|\boldsymbol{e}\setminus\{e\})}\right].$$

    (b) Set $\boldsymbol{e} := \boldsymbol{e}\setminus\{e'\}$.
(4) Until $\mathrm{conf}(\boldsymbol{e}) < t$.

It should be emphasized that as evidence is retracted during conflict resolution, we will in general need to perform standard probability updating when calculating the probabilities required for Step 3a and Step 4.

3. Empirical results

The proposed methodology has been tested on real-world data from a coal based power plant as well as simulated data from an oil production facility. In the latter case the data was generated based on a model that includes the dynamics of the facility as well as control loops.

3.1. Power plant data

We received data about the power plant under normal system operation with load average 90–100%, i.e., the power plant operated between 90% and 100% of its full capacity. The data set contains 9600 cases, and each case consists of 87 simultaneous observations

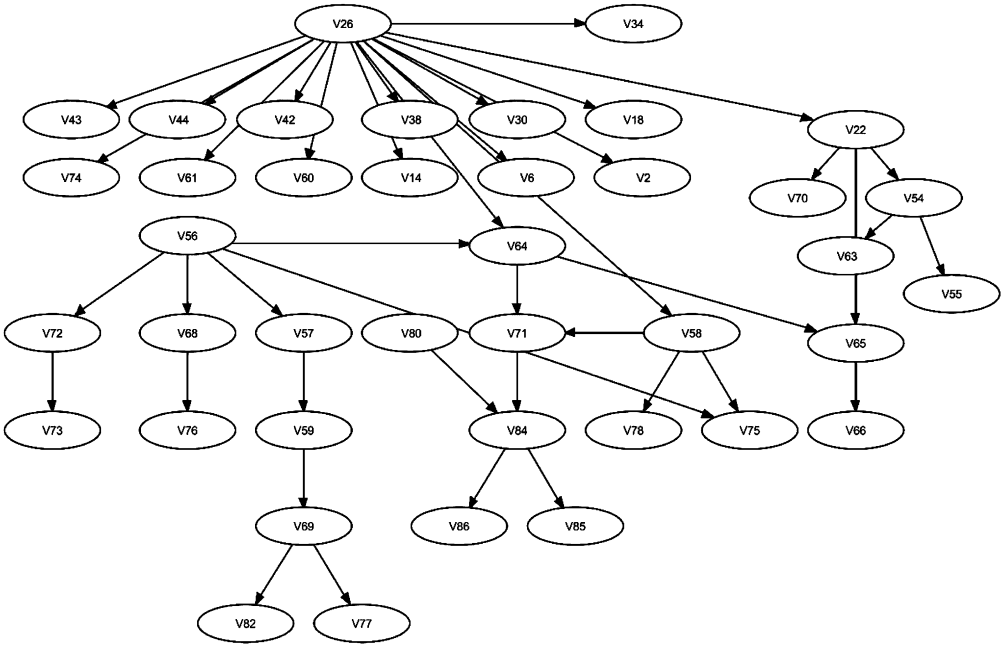

Fig. 2. The network learned from the power plant data.

with no missing values.[1] The power plant can roughly be seen as being composed of four distinct coal mills powering a turbine, thereby providing a natural partitioning of the observations into five disjoint sets: for each coal mill there are 12 distinct observations, and 35 observations summarize general properties of the power plant (the remaining 4 observations can be derived from the other observations, and they are therefore redundant). The cases do not only contain actual sensor values, but they also include soft sensors, i.e., artificial "sensors" that have been computed based on the values of other sensors, as well as set-points and other indirect signals. Unfortunately, as the domain experts could not provide information for distinguishing between the observations, they were all treated similarly.

As a preprocessing step, all data sets were naively discretized using equal width binning, where the number of bins were chosen (based on several tests) to be 3. Based on the preprocessed data, we learned a Bayesian network model for a single coal mill as described in Section 2.1; the actual learning was performed using the software tool PowerConstructor with a 0.1-threshold for the conditional independence tests [2,3]. In the learned model, six variables turned out to be completely independent of the other variables and they were therefore removed together with the redundant observations (see Fig. 2). It should be emphasized that the learned model is only used as a factorization of the joint probability distribution and should not be subject to interpretation from e.g., a causal point of view. One part of the model stands out, though, namely the node cluster consisting of V26

[1] Since each case contains sensors readings for a particular point in time, the database can also be interpreted as a sequence of "snapshots" of the plant.

together with its children. Node V26 represents the sensor measuring the water concentration in the coal, and the nodes that appear as children of V26 can, except for one node, be partitioned into two disjoint subsets with distinct semantics: 10 nodes correspond to sensors which either directly or indirectly measures the volume/pressure and temperature of the air that is used for drying the coal, and 4 nodes correspond to sensors related to the load average of the plant and which are therefore closely related to the water concentration in the coal.

Finally, it should be noted that since the database is complete, the parameters of the model could simply be estimated using frequency counts.

In addition to the data sets for normal system operation, we received three data sets that each contained 1441 cases. Two of the data sets covered actual errors/abnormal situations whereas the last represented an "unusual behavior" that it would be interesting to detect:

- The fall-pipe leading coal into the power plant becomes clogged.
- A temperature sensor becomes faulty.
- A large load change (from 60–75% to 90–100%) occurs while the water concentration in the coal is high, thereby making it difficult to regulate the production process.

We have tested the proposed methodology by simulating on-line performance using the "clogged fall-pipe" data set as well as the "faulty-sensor" data set. Both tests were performed "blind-folded", i.e., we first analyzed the data and then, *after* the analysis, we discussed our findings with the domain experts.

A plot of the conflict measures for the "clogged fall-pipe" data set is depicted in Fig. 3. From the plot we see that we have positive conflict measures from observation 1136 and forward, i.e., the conflict measures indicate that the system makes a transition from a normal to an abnormal system state at 1136. This is also consistent with the information provided to us, namely that the system entered an abnormal state (the fall-pipe became clogged) between 1100 and 1144. Another interesting aspect of the plot is the fluctuations

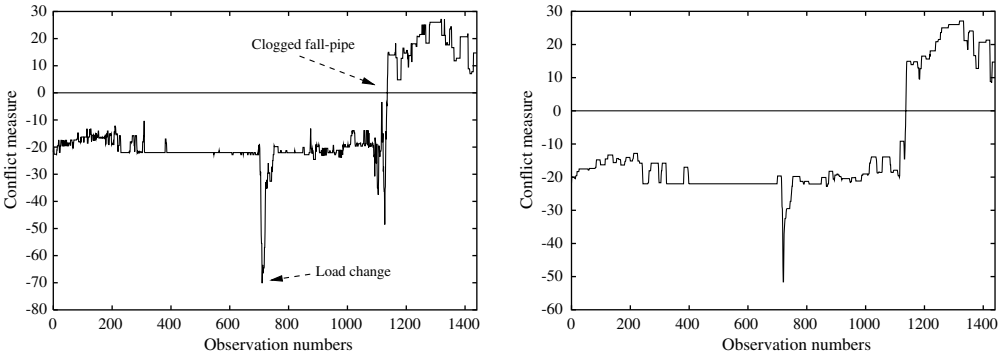

Fig. 3. The left-hand figure shows a plot of the conflict measure for each case in the "clogged fall-pipe" data set; a value above 0 indicates a conflict. Note how the conflict measure is affected by the load-change and the fall-pipe becoming clogged. To reduce the noise in the data, the right-hand figure shows the 0.9 percentile of the last 30 cases.

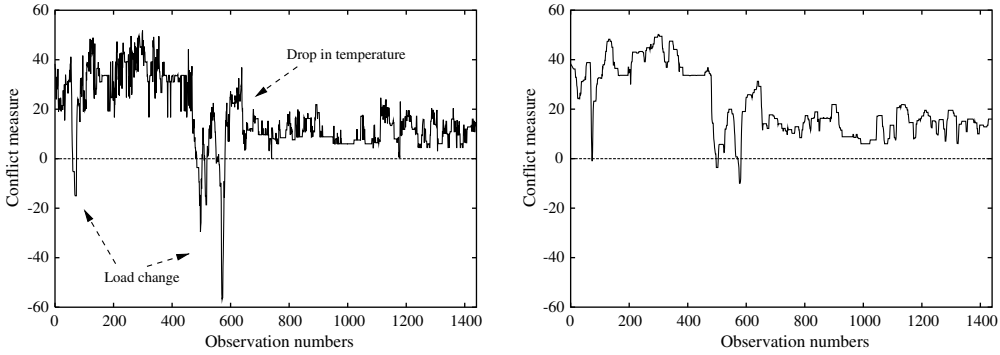

Fig. 4. A plot of the conflict measure for each case in the "faulty-sensor" data set; a value above 0 indicates a conflict. Note how the conflict measure is affected by the load-changes and the drop in temperature. The right-hand figure shows the 0.9 percentile of the last 30 cases.

in the conflict measure that appears around observation 700 and lasts until approximately 780. We were later told that in this interval the system actually made a short change in load average from 99% to 84% and then back again.

When performing conflict resolution, the algorithm indicates that the sensor measuring the water-percentage in the coal can explain all the conflicts. Ideally, we would have liked the system to pinpoint that the fall-pipe is clogged. However, since this is not part of the vocabulary provided by the model we interpret the result as indicating that there is an inconsistency in the energy balance of the system, and that this inconsistency can best be explained by the water percentage in the coal; this was also consistent with the analysis by the engineers.

A similar test was made on the "faulty sensor" data set, where the conflict measures can be seen in Fig. 4. As suggested by the plot, the conflict measure indicates that the system entered the abnormal state prior to the first observation; this was later confirmed by the engineers. We were also informed that in the beginning of the data set and around observation 600, there were two quick changes in the load averages (from 90–100% to 80% and back again); these changes are reflected as quick changes in the calculated conflict measures. Finally, we were told that around observation 600 the temperature drops from 100 °C to 90 °C (at which level it stays for the remaining observations). Observe, that around this observation we also see a permanent drop in the conflict measure that seems to stabilize around observation 1100.

When performing conflict resolution we found that after observation 600, there were six significant sensors that could explain the conflict. We were informed that four of the sensors were actually significant for this scenario, but that the other two "sensors" should not have been picked out since they were set-points rather than sensors. However, the identification of these sensors actually makes sense as there is a conflict between the system sensors and the set-points.

Finally, we have made a tentative analysis of the "production regulation" data set. A difficulty with this data set is that the learned model only covers normal operation during load average 90–100%. Hence, we have only considered the observations made after the load change has been completed. For the resulting data set the distinguishing characteristic is that the coal has a high water concentration, which made it difficult to regulate the

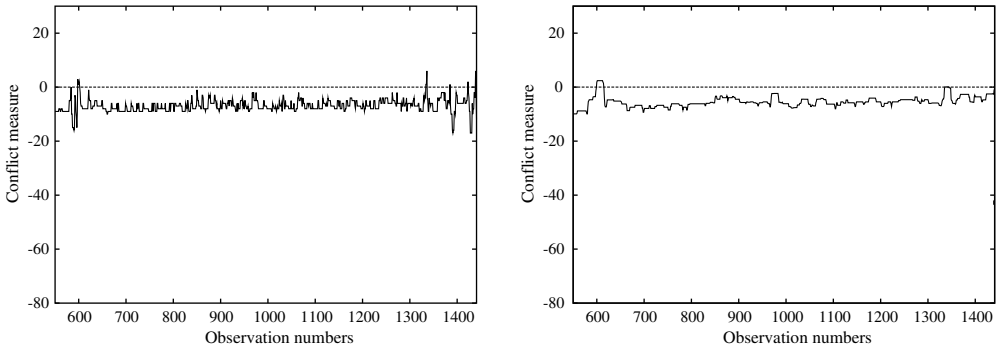

Fig. 5. A plot of the conflict measure for the "production regulation" data set after the change has taken effect. The system is correctly classified as *not* being in an abnormal state. The right-hand figure shows the 0.9 percentile of the last 30 cases.

production process.[2] That is, the data set has *not* been produced from a system state which should be classified as being abnormal, but rather an unusual system state that it would be interesting to detect (in case it would eventually result in an abnormal state). Fig. 5 shows a plot of the conflicts after observation 550 where the load change has been completed.

As can be seen from the figure, the conflict values are all below 0 (except for a few single cases). This is consistent with the system not being in an abnormal state. However, from the measurements we can also see that the average conflict value is higher than for normal operation: For the "production regulation" data set, the average conflict value is -7.44, but during normal operation in the "clogged-fall-pipe" data set the average conflict value is between -10.34 and -22.8 with an average of -19.96. That is, you may be able to discriminate between different types of normal system operation by also considering the value of the conflict measure and not only whether it is positive or negative.

3.2. Oil production data

We have received a database with 10,000 simulated cases for normal system operation for an oil production facility; each case in the database covers 140 sensors with white noise added to the sensor values.[3] The database was generated from a temporal causal model, which also simulated standard process variations. Hence, the database shares the same characteristics w.r.t. learning as the power plant database (see Section 2.1). All of the sensor values appeared as real-valued output, so as a preprocessing step all variables/sensors were discretized. The actual discretization was performed using cross-validation to find the number of bins (with a maximum of 5) that maximizes the estimated likelihood of the data; the actual discretization was performed using Weka [22].

In order to test the proposed methodology in this setting, we used two other data sets both containing 10,000 cases. The first data set was generated by simulating faults in the pumping system whereas the second data set was generated by simulating faults in the cooling system (see also Table 2).

[2] The actual difficulty is caused by the system not being able to dry the coal during a load average of 90–100%.

[3] Similar to the power plant database, the database can be interpreted as a sequence of "snapshots" of the facility.

Table 2
The table summarizes the changes in the production process for the "Pump" data set and the "Cooling" data set, respectively. Note: The changes in the two scenarios are initiated at the same points in time

Time	"Pump" data set	"Cooling" data set
30	Small leak in the pump	Small external leak in the cooling system
1500	Large leak in the pump	Large external leak in the cooling system
3000	Normal operation	Normal operation
3500	Small degradation of motor efficiency	Small internal leak in the cooling system
5000	Large degradation of motor efficiency	Large internal leak in the cooling system
6500	Normal operation	Normal operation
7000	Small degradation of pump efficiency	Moderate fouling
8500	Large degradation of pump efficiency	Significant fouling

A plot of the conflict measure for the "Pump" data set is shown in Fig. 6(a). As in the previous section, Fig. 6(b) shows the 0.9-percentile over the last 30 cases. The vertical lines in the two plots correspond to the points in time where changes are initiated (see Table 2). As can be seen from Fig. 6, there are significant changes in the conflict measure at time 1500, 3000, 5000, 6500 and 8500, which either correspond to large errors in system operation or changes back to normal system operation. From Table 2 we see that the changes appearing at 30, 3500 and 7000 correspond to small errors in the system operation and, accordingly, they are also less apparent in the plots. In particular, the change appearing at 3500 occurs before the system has settled into stationary normal system operation.

A similar plot of the conflict measure for the "Cooling" data set is given in Fig. 7(a). Analogously to the previous data set, there is a significant change in the conflict measure for all errors except at time 30, 3500 and 7000.

Note that the conflict measures for both databases are all negative, which is a consequence of the decomposition property (Proposition 2.1) as discussed in Section 2.2. Thus, in order to perform conflict resolution the conflict threshold (see Algorithm 2.1) should be specified based on the values observed during normal operation. Moreover, in order to detect changes in system operation we need to track jumps in the conflict measure.

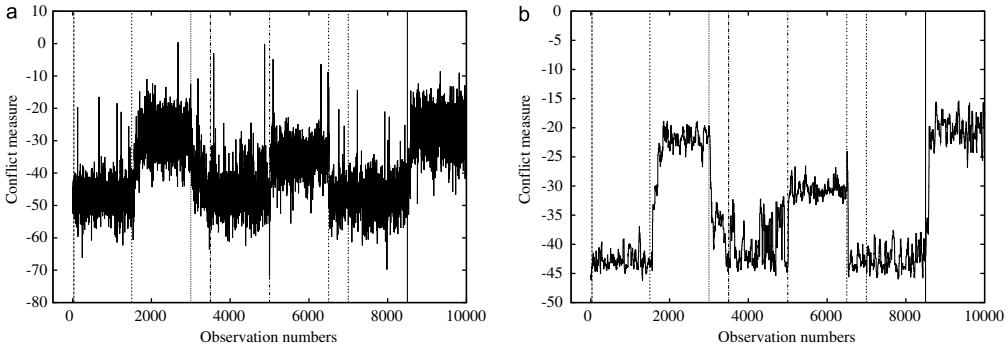

Fig. 6. The left-hand figure shows a plot of the conflict measure for each case in the "Pump" data set. The vertical lines indicate when a change in the production process is initiated as specified in Table 2. The figure to the right shows the 0.9 percentile of the last 30 cases.

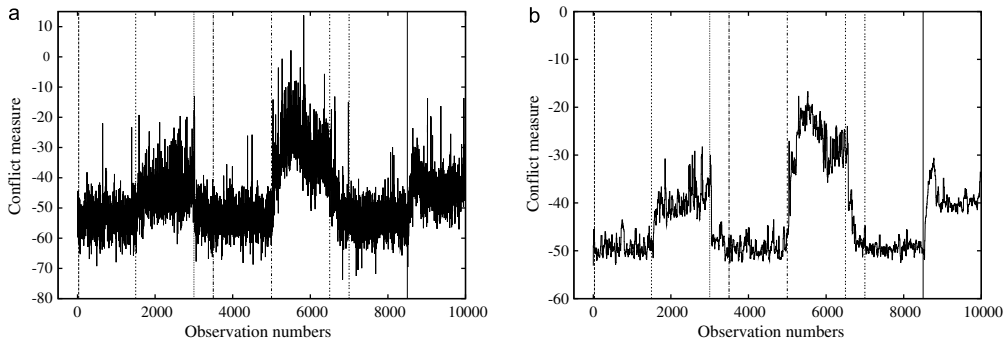

Fig. 7. The figure to the left shows a plot of the conflict measure for each case in the "Cooling" data set. The vertical line indicates when a change in the production process is initiated as specified in Table 2. The right-hand figure shows the 0.9 percentile of the last 30 cases.

3.3. Change point detection

In order to track the changes in the conflict measure we have applied a method related to the work by Yamanishi and TakeUchi [23], and which can be seen as an extension of the method presented in Section 2.2. Specifically, we assume that under normal system operation the conflict values can (approximately) be seen as independent samples from a normal distribution with fixed mean and variance. Under this assumption we can model the lth conflict value as a random variable with a normal distribution, f, where the mean μ_l and the variance σ_l^2 are estimated as the sample mean and sample variance of the last m observations (also called the *relevant history*):

$$\hat{\mu}_l = \bar{x}_{l,m} = \frac{1}{m}\sum_{j=1}^{m} x_{l-j}, \quad \hat{\sigma}_l^2 = \frac{1}{m}\sum_{j=1}^{m}(x_{l-j} - \bar{x}_{l,m})^2.$$

Thus, an immediate approach for detecting a change point could be to calculate the logarithmic loss for the last n observations $(x_{i+1}, \ldots, x_{i+n})$, and then raise an alert in case the value is above a predefined threshold:

$$-\log_2 f(x_{i+1}, \ldots, x_{i+n} | x_1, \ldots, x_i) = -\sum_{j=1}^{n} \log[f(x_{i+j} | \hat{\mu}_{i+j}, \hat{\sigma}_{i+j}^2)] > \delta.$$

This approach, however, is very sensitive to fluctuations in the conflict value (thereby producing false alerts), and it has a difficulty in detecting drifts in the conflict measure. To alleviate these two problems, we instead compare the model above with another model f', where the mean and the variance are estimated without taking the last s observations into account. That is, the relevant history, used to estimate μ and σ^2, is shifted s observations back:

$$\hat{\mu}_i' = \frac{1}{m}\sum_{j=1}^{m} x_{i-j-s}, \quad \hat{\sigma}_i'^2 = \frac{1}{m}\sum_{j=1}^{m}(x_{i-j-s} - \bar{x}_{i-s,m})^2.$$

The models are compared by evaluating the difference in log-likelihood of the models given the last n observations (also called the *score difference*):

$$\Delta_{f,f'}(x_{i+1},\ldots x_{i+n}) = \log_2\left[\frac{f(x_{i+1},\ldots,x_{i+n}|\hat{\mu}_i,\hat{\sigma}_i^2)}{f'(x_{i+1},\ldots,x_{i+n}|\hat{\mu}_i',\hat{\sigma}_i'^2)}\right].$$

Note that $\Delta_{f,f'}$ should not be considered a conflict measure as in Section 2.2, but rather a score for comparing two competing models for normal operation. In particular, during normal operation we would expect the score to be within an interval $[-\delta{:}\delta]$, and for actual change points we would expect $\Delta > \delta$. The value of δ influences the ratio of false positives and false negatives and could, e.g., be chosen based on the conflict values observed during normal/stable system operation. The value of n determines the response time of the system, but by choosing the value of n too small we risk having the score dominated by random fluctuations.

To test the method, we first simulated on-line fault detection using the conflict values produced by the "clogged fall-pipe" data set. A plot of $\Delta_{f,f'}$ using $n = 5$ (the number of conflict values used to compare the models) $m = 5$ (the size of the relevant history) and $s = 20$ (the number of conflict values ignored in the second model) is shown in Fig. 8(a). Fig. 8(b) shows another example with $n = m = 10$ and $s = 40$. In both plots we have an increase in $\Delta_{f,f'}$ when the load-average changes (around observation 600) and when the fall-pipe becomes clogged (around observation 1100). The changes in $\Delta_{f,f'}$ appearing around observation 200 are caused by the relative large fluctuations in the conflict values occurring after intervals with zero variance. Unfortunately we have not been able to identify the source of these fluctuations.

We have performed similar tests for the "faulty sensor" data set as shown in Fig. 9. As described previously, for this data set the system had entered an abnormal system state already prior to the first observation. This is also reflected in the score difference, which is less stable than for the "clogged fall-pipe" data set. Most significantly, though, are the two peaks corresponding to the two changes in load average, as well as the peaks that occur around observation 1100, where the system appears to stabilize after the temperature drop.

For the oil production data, the detection of change points is obscured by the relative large amount of noise in the system. Fig. 10 shows the score difference for the "Pump" data set using ($m = 15$, $n = 15$, $s = 45$) and ($n = 20$, $m = 20$, $s = 50$), respectively. In the first test, the change points caused by the small degradations in the production process

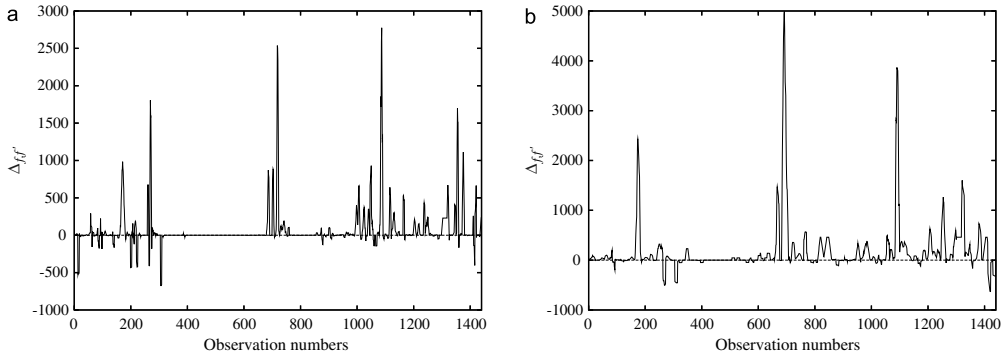

Fig. 8. The left-hand figure shows a plot of $\Delta_{f,f'}$ (with $n = m = 5$ and $s = 20$) for the conflict values produced by the "clogged fall-pipe" data set. The right-hand figure shows a similar plot, but with $n = m = 10$ and $s = 40$.

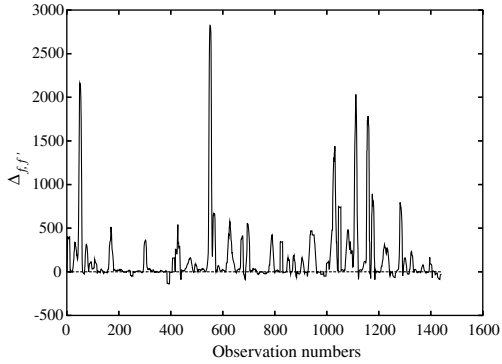

Fig. 9. The figure shows the score difference for the faulty sensor data set using $m = n = 10$ and $s = 25$.

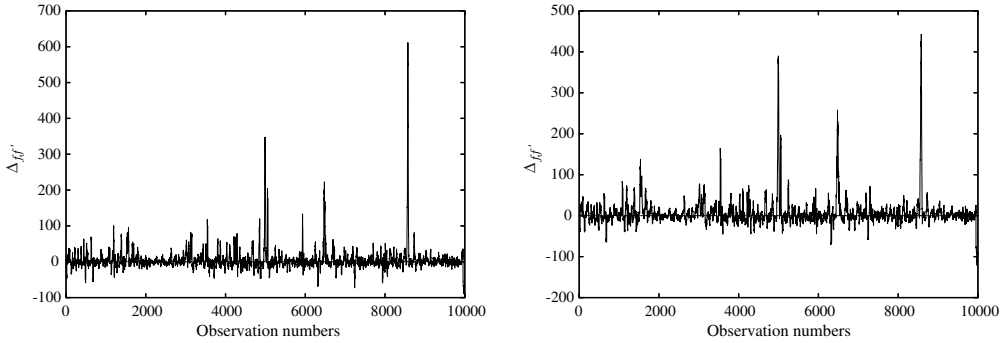

Fig. 10. The left-hand figure shows a plot of $\Delta_{f,f'}$ for the conflict measure produced by the "Pump" data set with $n = m = 15$ and $s = 45$. The right-hand figure shows a similar plot, but with $n = m = 20$ and $s = 50$.

are less apparent as compared to the second test, where we increase the relevant history and the number of discarded samples. Our experiments indicate that the parameter values governing change point detection have a significant impact on the score difference as well as the response time of the system. However, a more formal analysis is outside the scope of this paper.

4. Conclusion and future work

We have proposed an alert system methodology based on conflict analysis. A distinguishing characteristic of the proposed methodology is that it only relies on a model for normal system operation, i.e., knowledge about the possible faults is not required. Moreover, the computational complexity of the algorithm ensures that on-line analysis is feasible. The methodology has been successfully tested on both real-world data from a power plant and simulated data from an oil production facility.

As part of ongoing research and future work, we are working on establishing alternative straw models in order to perform a more refined conflict analysis; see also the discussion in [16,15] concerning the independence straw model [14]. Having an alternative straw

model might also reduce the effect of the decomposition property. That is, when faulty sensors' impact on the conflict measure is dominated by strongly correlated sensors. Moreover, it could also be interesting to consider alternative types of models for normal operation. For example, instead of looking for a general Bayesian network model, which tends to become very dense, we might focus on a restricted subclass of models, such as the Naïve Bayes models [16]. In addition, rather than discretizing the sensor values one could also try to learn a model with continuous variables using e.g., mixtures of truncated exponentials [21].

In the process of finishing the paper, we became aware of the work recently published by Ibargüengoytia et al. [12]. Ibargüengoytia et al. [12] take outset in the related field of sensor validation, where they use Bayesian networks for representing normal sensor operation. However, instead of using conflict measures for detecting faults they compare the actual sensor readings with the expected sensor readings; a discrepancy between these values is then interpreted as a possible fault. A more thorough comparison is outside the scope of the present paper and is subject to future research.

Acknowledgements

We would like to thank Rasmus Madsen and Babak Mataji from ELSAM engineering for providing us with data from the power plant. We would also like to thank John-Morten Godhavn from Statoil ASA for supplying us with data from the oil production facility, and Erling Lunde from Dynamica AS for helpful comments regarding the technical layout of the facility. In particular, we would like to thank Helge Langseth for valuable discussions and comments, and *Hugin Expert* (www.hugin.com) for giving us access to the *Hugin Decision Engine* that forms the basis of our implementation. Finally, we would like to thank the anonymous reviewers for their constructive comments and suggestions for improving the paper.

References

[1] X. Boyen, N. Friedman, D. Koller, Discovering the hidden structure of complex dynamic systems, in: K.B. Laskey, H. Prade (Eds.), Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, 1999, pp. 91–100.

[2] J. Cheng, D.A. Bell, W. Liu, Learning belief networks from data: an information theory based approach, in: Proceedings of the Sixth ACM International Conference on Information and Knowledge Management, 1997, pp. 325–331.

[3] J. Cheng, R. Greiner, J. Kelly, D. Bell, W. Liu, Learning Bayesian networks from data: an information-theory based approach, Artificial Intelligence 137 (1–2) (2002) 43–90.

[4] C.-F. Chien, S.-L. Chen, Y.-S. Lin, Using Bayesian network for fault location on distribution feeder, IEEE Transactions on Power Delivery 17 (3) (2002) 785–793.

[5] G.F. Cooper, E. Herskovits, A Bayesian method for constructing Bayesian belief networks from databases, in: B.D. D'Ambrosio, P. Smets, P.P. Bonissone (Eds.), Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence, 1991, pp. 86–94.

[6] A. Darwiche, A differential approach to inference in Bayesian networks, Journal of the ACM 50 (3) (2003) 280–305.

[7] A.P. Dawid, Applications of a general propagation algorithm for a probabilistic expert system, Statistics and Computing 2 (1992) 25–36.

[8] J. de Kleer, J. Kurien, Fundamentals of model-based diagnosis, in: Proceedings of the Fifth IFAC Symposium on Fault Detection, Supervision, and Safety of Technical Processes (Safeprocess), 2003, pp. 25–36.

[9] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, John Wiley & Sons, New York, 1973.
[10] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Machine Learning 29 (2–3) (1997) 131–163.
[11] N. Friedman, K. Murphy, S. Russell, Learning the structure of dynamic probabilistic networks, in: G.F. Cooper, S. Moral (Eds.), Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, 1998, pp. 139–147.
[12] P.H. Ibargüengoytia, S. Vadera, L.E. Sucar, A probabilistic model for information and sensor validation, The Computer Journal 49 (1) (2006) 113–126.
[13] F.V. Jensen, Bayesian Networks and Decision Graphs, Springer-Verlag, New York, 2001, ISBN 0-387-95259-4.
[14] F.V. Jensen, B. Chamberlain, T. Nordahl, F. Jensen, Analysis in hugin of data conflict, in: Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence, 1990Also published in Uncertainty in AI 6, 519–528, North-Holland, Amsterdam, 1991.
[15] Y.-G. Kim, M. Valtorta, On the detection of conflicts in diagnostic Bayesian networks using abstraction, in: P. Besnard, S. Hanks (Eds.), Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, 1995, pp. 362–367.
[16] K.B. Laskey, Conflict and surprise: heuristics for model revision, in: B.D. D'Ambrosio, P. Smets, P.P. Bonissone (Eds.), Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, 1991, pp. 197–204;
D. Lowd, P. Domingos, Naive Bayes models for probability estimation. in: Proceedings of the Twentysecond International Conference on Machine Learning, 2005, pp. 529–536.
[17] A.L. Madsen, F.V. Jensen, Lazy propagation: a junction tree inference algorithm based on lazy evaluation, Artificial Intelligence 113 (1999) 203–245.
[18] N. Mehranbod, M. Soroush, M. Piovoso, B.A. Ogunnaike, Probabilistic model for sensor fault detection and identification, American Institute for Chemical Engineers Journal 49 (7) (2003) 1787–1802.
[19] J. Pearl, Probabilistic Reasoning in Intelligent SystemsRepresentation and Reasoning, Morgan Kaufmann Publishers, San Mateo California, 1988, ISBN 0-934613-73-7.
[20] R. Reiter, A theory of diagnosis from first principles, Artificial Intelligence 32 (1) (1987) 57–95.
[21] V. Romero, R. Rumí, A. Salmerón, Structural learning of Bayesian networks with mixtures of truncated exponentials, in: Proceedings of the Second European Workshop on Probabilistic Graphical Models, 2004, pp. 177–184.
[22] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools with Java Implementations, Morgan Kaufmann, San Francisco, 2000, version 3.4.3.
[23] K. Yamanishi, J.-I. TakeUchi, A unifying framework for detecting outliers and change points from non-stationary time series data, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, 2002, pp. 676–681.